VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Mobilní aplikace pro správu osobních financí

Mobile Application for Personal Finance Management

2016 **Pavel Tížek**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Pavel Tížek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Mobilní aplikace pro správu osobních financí
Mobile Application for Personal Finance Management

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je navrhnout a implementovat mobilní aplikaci pro správu osobních financí pro platformu Android. Bude implementována možnost správy více účtů, správa kategorií záznamů, export a import dat prostřednictvím vhodných formátů (ABO, JSON, apod.) a využití pokročilých metod prezentace záznamů a vývoje stavu účtů.

1. Prostudujte existující elektronické aplikace pro správu osobních financí.
2. Zaměřte se na specifika vyplývající z českého prostředí (ABO formát, sdílené účty, apod.) a popište je.
3. Analyzujte požadavky, které budou kladeny na aplikaci a navrhněte vhodné uživatelské rozhraní včetně grafické reprezentace potřebných údajů.
4. Implementujte aplikaci a případnou serverovou část, bude-li potřeba.
5. Otestujte funkčnost aplikace a vyhodnoťte dosažené výsledky, posuďte vhodnost implementace pro jednotlivé potenciální uživatele.

Seznam doporučené odborné literatury:

[1] Steele, J., To, N.: The Android Developer's Cookbook: Building Applications with the Android SDK, Addison-Wesley Professional, 2010, ISBN-13: 978-0321741233.
[2] Meier, R.: Professional Android 4 Application Development, Wrox, 2012, ISBN-13: 978-1118102275
[3] Hashimi, S.: Pro Android 2, Apress, 2010, ISBN-13: 978-1430226598
[4] Technický popis struktury ABO formátu pro programátory. [online] [cit. 2015-09-15] Dostupné z: <http://www.csas.cz/static_internet/cs/Obchodni_informace-Produkty/Prime_bankovnictvi/Spolecne/Prilohy/ABO_format.pdf>
[5] Klientský formát ABO. [online] [cit. 2015-09-15] Dostupné z: <http://www.mojebanka.cz/file/cs/bdsk_format_abo_sk.pdf>
[6] FIO API bankovnictví. [online] [cit. 2015-09-15] Dostupné z: <http://www.fio.cz/docs/cz/API_Bankovnictvi.pdf>

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Moravec, Ph.D.**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *29. dubna 2016*

.....

podpis studenta

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 29. dubna 2016

.....

podpis studenta

Poděkování

Rád bych poděkoval Ing. Pavlu Moravcovi, Ph.D. za pomoc a konzultaci při vytváření této bakalářské práce a svým nejbližším za poskytnutí podpory a zázemí při psaní této práce.

Abstrakt

Cílem této bakalářské práce je prostudovat problematiku vývoje aplikací pro operační systém Android, analyzovat problematiku správy domácích financí a vytvořit mobilní aplikaci pro správu osobních financí, která uživateli umožní evidovat finance v zařízeních podporující platformu Android. Aplikace má umožnit správu více účtů a kategorií, jednoduše a přehledně zobrazit data ve formě grafů. Dále má umožnit import dat z internetového bankovnictví. Uživatel si může zobrazit graficky historii stavu účtu ve zvoleném období.

Klíčová slova: Android, aplikace, finance

Abstract

The purpose of this bachelor thesis is to study the problems of developing applications for the Android operating system, analyze the problems of management of household finances and develop a mobile application for managing personal finances that allows users to register finances on devices supporting Android platform. The application has to permit management of multiple accounts and categories, easily and clearly display the data in graphs. Also allow import of data from internet banking. The user can graphically display the history of the account balance in the selected period.

Key words: Android, application, finance

Obsah

Seznam použitých zkratek

ABO	Formát pro import a export dat z bank
ADB	Android Debug Bridge
API	Application Programming Interface
AVD	Android Virtual Device
DDMS	Dalvik Debug Monitor Server
IDE	Integrated Development Enviroment, Vývojové prostředí
JSON	Datový formát, JavaScript Object Notation
ORM	Object Relation Mapping
XML	Extensible Markup Language

Seznam obrázků a seznam tabulek

Úvod

V dnešní době, kdy lidé počítají každou korunu, je důležité mít přehled o penězích. Vždy se hodí vědět kolik a za co člověk utrácí a jestli nejsou výdaje větší než příjmy. Finance lze sledovat pomocí internetového bankovnictví nebo výpisů z banky v papírové podobě. Nevýhodou tohoto způsobu je nutnost uschovat si všechny záznamy a ručně si vše zapisovat. Další nevýhodou ručního zapisování je například nesprávné přepisování dat. Kdokoli může při sčítání udělat chybu a tím můžou nesprávně hospodařit s penězi. Tento problém mobilní aplikace vyruší, jelikož vše si dokáže sama přepočítat.
Existující mobilní i webové aplikace, které nabízí jednoduchou správu financí a umožňují tak pohodlné ukládání a zobrazení dat. Neumožňují však nahrát uskutečněné platby z internetového bankovnictví, proto je nutné všechny zadat do aplikace ručně. Mnoho bank umožňují export výpisu z účtů. Některé banky používají datový formát ABO, který je standardizovaný a je jednoduché ho strojově zpracovat. Některé banky ale tento formát nepodporují a nabízí export v nestandardizovaných formátech jako CSV, XML nebo TXT.
Proto jsem se rozhodl vytvořit aplikaci, která uživateli umožní spravovat finance v telefonu a bez zapisování na papír či do bloků. Uživatelé si v této mobilní aplikaci budou moci zobrazit i grafy, které ukážou, za co nejvíce utratí nebo celkové příjmy a výdaje. Záleží jen na uživateli, který graf chce zobrazit. Může si také zaznamenat přesnou polohu platebního místa s detailním popisem.
Bakalářská práce je členěna do 7 základních kapitol. Jako každá práce má i tato jistý teoretický začátek, první kapitolu, kde se zabývám platformou Android.
Každý projekt se musí nejprve analyzovat. Musíme vytyčit pro koho je aplikace určena, jak bude fungovat a co by bylo vhodné přidat pro inovaci do nové aplikace. O tomto pojednává druhá kapitola. Další postup při vývoji je zjištění situace na trhu. Což znamená, najít podobné aplikace zabývající se správou financí. Součástí třetí kapitoly je porovnání ostatních aplikací pro platformu Android. Čtvrtá kapitola se věnuje návrhu aplikace. Zde jsem se zaměřil na výběr způsobu ukládání dat, samotný návrh databáze a výběru nejnižší podporované verze Android API. Nejdůležitější část, pátá, se zabývá samotnou implementací. Jsou zde uvedeny všechny použité komponenty, popis chování aplikace při importování dat a vkládání nových plateb. O tom jestli je aplikace hotova, či správně naprogramována nám ukáže samotné testování, které je popsáno v šesté kapitole. To je prováděno jak na virtuálním zařízení, tak i na mobilním zařízení. V poslední kapitole je ukázáno uživatelské rozhraní a vzhled aplikace.

1 Platforma Android

Android je operační systém pro mobilní telefony založený na jádře Linuxu a vychází z open source software (volně dostupný zdrojový kód). V současnosti se používá pro mnoho zařízení, například pro mobilní telefony, tablety, televize, chytré hodinky a mnoho dalších.

1.1 Historie

Počátky platformy Android sahají do roku 2003, kdy vznikla stejnojmenná firma. V roce 2005 tuto společnost odkoupil Google a udělal z ní svou dceřinou společnost. V roce 2007 vytvořili konsorcium Open Handset Alliance, a následně představili nový mobilní operační systém Android [2]. První telefon s platformou Android, T-Mobile G1 byl uveden na trh roku 2008 ve Spojených státech amerických a rok poté byl představen tento telefon i v České republice.
Od této doby Android získával na popularitě hlavně díky jeho otevřenosti a velké nabídky zařízení. Dnes Android pohání více než 80 % mobilních telefonů[1].

1.2 Běhové prostředí

K vývoji aplikací pro Android můžeme vybrat z několika možných běhových prostředí. Každé z nich nabízí své výhody i nevýhody. Níže je popsáno využití jednotlivých technologií.

1.2.1 Java

Java je jedním z nejrozšířenějších programovacích jazyků. Je využíván především kvůli velkým množstvím knihoven a velmi pokročilými vývojovými prostředími. Oproti aplikacím psaných v jazyce C/C++ jsou náročnější na výkon zařízení. Je odolnější proti vývojářským chybám, a tudíž napadení zařízení je těžší.
Verze jazyka Java pro počítače a pro Android je mírně odlišná. Při vytváření aplikace se používá virtuální stroj Dalvik, přičemž v počítačové verzi se používá Java Virtual Machine. Kód se nejdříve zkompiluje a uloží na disk a při spuštění aplikace se následně nahraje do paměti. Díky tomuto kompilování dosahují aplikace mnohem vyššího výkonu [4].

1.2.2 HTML 5

Toto prostředí je ideální pro začátečníky ve vývoji aplikací. Lze použít tam, kde není potřebný vysoký výkon aplikace. Můžeme využít webovou aplikaci, která umožní uložit potřebné prostředky do mezipaměti a poté je využít v případě nedostupnosti internetového připojení [5].
Nejedná se o aplikace dostupnou z běžného katalogu, jde jen o odkaz na webovou stránku. Tento odkaz si musí uživatel vložit do záložek nebo umístit odkaz na aplikaci na domovskou obrazovku. HTML 5 nabízí velice rychlé vytvoření aplikace a to pro různé operační systémy. Toto prostředí nespolupracuje s rozhraním Bluetooth a neumožňuje spustit jinou aplikaci.

1.2.3 C/C++

Jazyky C a C++ jsou vhodné pro aplikace, které vyžadují vysoký výpočetní výkon. Nejčastěji je tedy používán pro 3D hry a složitější výpočetní aplikace [6].

1.2.4 Xamarin

Tato vývojová platforma umožňuje vyvíjet pro platformy Android, iOS a Windows Phone. Xamarin. Využívá jazyk C# a z něj vytváří nativní kód pro zmiňované platformy [7]. Podpora vývoje v Microsoft Visual Studio a Xamarin Studio.

1.3 Vývojové prostředí

Dříve nejpoužívanější vývojové prostředí Eclipse s ADT pluginem bylo postupně nahrazeno novějším Android Studiem. Dnes už se převážně používá Android Studio, původní Eclipse už

není nadále pro vývoj Android aplikací podporováno. Toto vývojové prostředí nabízí mnoho různých nástrojů, které nám slouží k usnadnění, ale také k urychlení práce při vývoji aplikací. Také nám nabízí nástroj, který slouží ke snadnějšímu vytvoření uživatelského rozhraní. Vývojáři také přidali editor pro vytváření překladů, který je velmi důležitý při tvorbě lokalizace aplikace. Od verze 2.0 také nabízí tzv. instant run, který umožňuje v některých případech nekompilovat celou aplikaci, ale jen změněné části. To je vhodné pro rychlé úpravy uživatelského rozhraní a malé změny kódu. Android Studio vychází z platformy IntelliJ.

1.4 Knihovny

Platforma Android (stejně jako většina platforem) dovoluje při vývoji aplikace používat knihovny třetích stran. Knihovny jsou malé moduly, jejichž použitím můžeme rapidně zrychlit vývoj samotné aplikace a zaměřit se více na jádro aplikace. V dnešní době existuje nepřeberné množství knihoven s různorodým zaměřením. Existují knihovny například pro práci s databází, komponenty aplikace (dialogy, navigation drawer) nebo pro síťovou komunikaci.
Velmi důležité (ne-li nezbytné pro vývoj aplikací v dnešní době) knihovny poskytuje firma Google Inc. Jednou z nich je knihovna Support Library[8] , která umožňuje použít komponenty z nejnovějších Android API i na starších verzích operačního systému Android využívajících nižší verze Android API.

1.5 Základní prvky aplikace pro Android

1.5.1 Aktivita

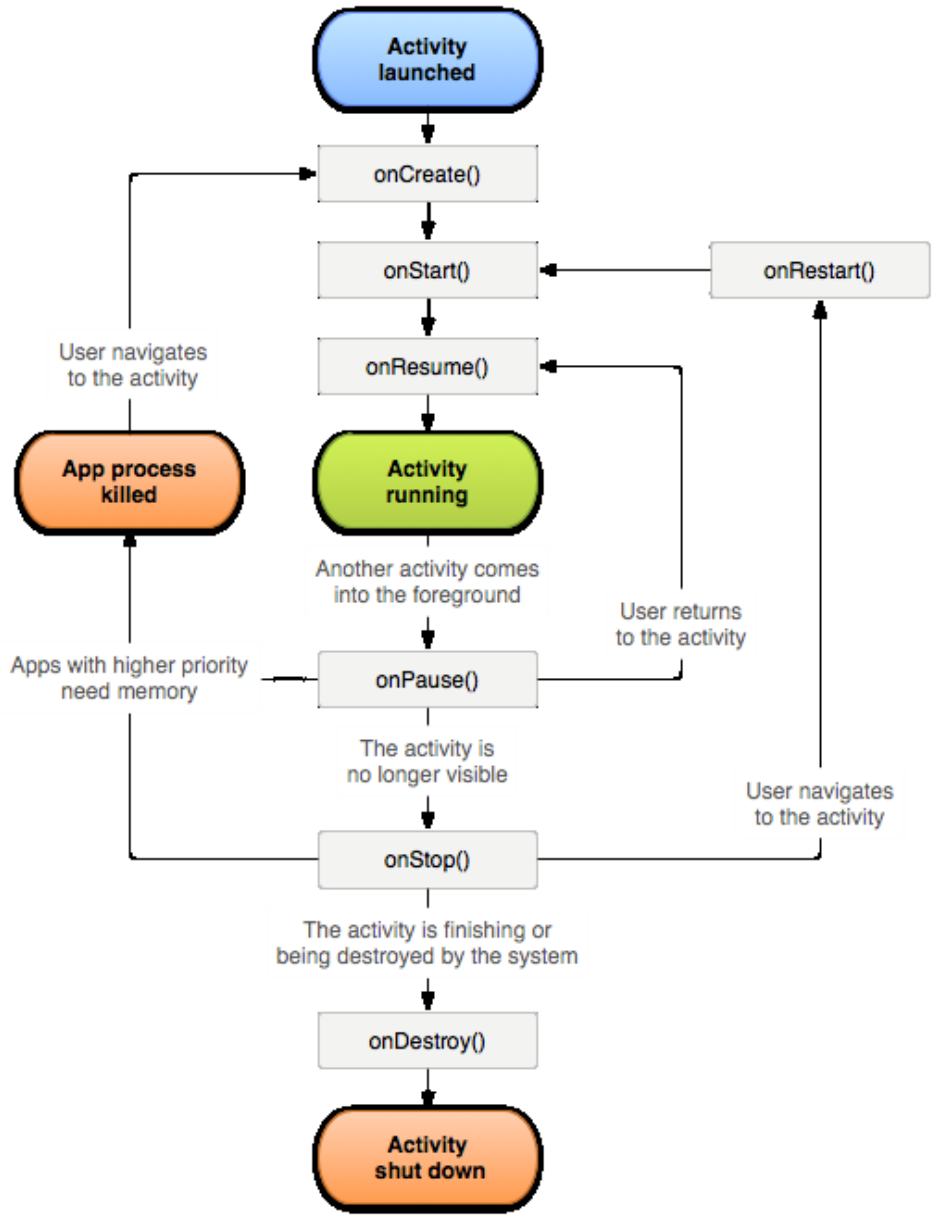
Aktivity jsou nezávislé třídy dědící z třídy android.app.Activity (ta dále dědí z třídy android.content.Context). Aktivita představuje jednotlivou obrazovku. Každá aktivita má své uživatelské rozhraní, které může vyplnit celou obrazovku, nebo jen její část. Z každé aktivity můžeme spustit další aktivitu a předat jí požadovaná data, přičemž původní aktivita je pozastavena. To znamená, že spuštěná aktivita je vždy viditelná.

Ukázka spuštění nové aktivity:

```
Intent intent = new Intent(getApplicationContext(),
NewPaymentActivity.class);
intent.putExtra("payment", payment);
startActivityForResult(intent, EDIT_TRANSFER_RQUEST_CODE);
```

Pro spuštění aktivity je zapotřebí vytvořit intent a do něj vložit data, která chceme předat spouštěné aktivitě.
Ve spouštěné aktivitě lze data získat následovným způsobem:

```
Payment payment = (Payment)
getIntent().getSerializableExtra("payment");
```

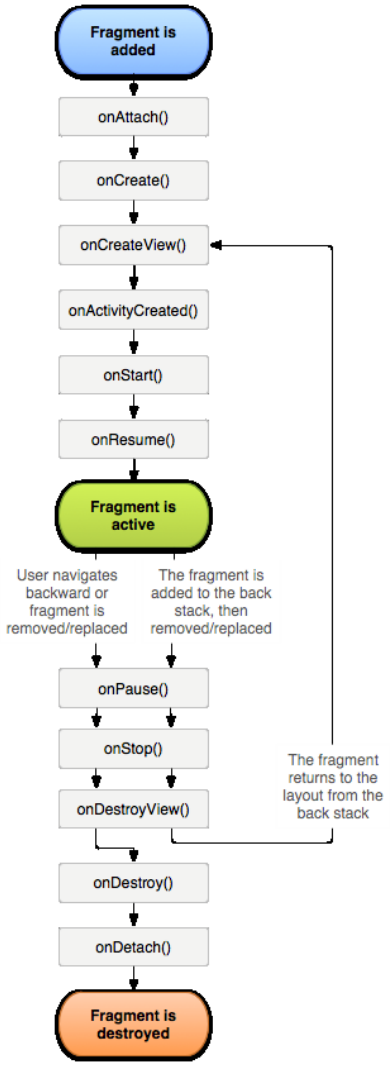


Obrázek 1: Životní cyklus aktivity, dostupné z [23]

Životní cyklus aktivity je rozdělen do třech stavů: spuštěná, pozastavená a zastavená. Na obrázku výše je znázorněn diagram všech tří stavů a jejich vzájemné přechody. Spuštěná aktivita je vždy viditelná a reaguje na interakci s uživatelem. Pozastavená aktivita je většinou částečně viditelná a je udržovaná v paměti. Typickým příkladem je aktivita překrytá dialogovým oknem. Aktivita samotná již nereaguje na uživatelský vstup a čeká na ukončení překrývající aktivity. Systém při nedostatku paměti může pozastavenou aktivitu ukončit. Zastavená aktivita je kompletně překrytá jinou a udržovaná v zásobníku. Při nedostatku systém nejdříve ukončí aktivity zastavené a poté aktivity pozastavené.

1.5.2 Fragment

Fragmenty slouží k zobrazení více částí aplikace na jedné obrazovce. Díky fragmentům můžeme rozdělit obrazovku na více části při použití jedné aktivity. To je užitečné zejména pro vývoj aplikací pro mobilní telefony a tablety zároveň. Díky fragmentům není potřeba pro každou obrazovku vytvářet novou aktivitu. Je tedy možné vytvořit jednoduchou aplikaci s jednou aktivitou a mnoha fragmenty.



Obrázek 2: Životní cyklus fragmentu, převzato z [24]

Stejně jako aktivita i fragment má tři základní stavy: spuštěný, pozastavený a zastavený. Tyto tři stavy korelují se stavy aktivity a metody fragmentu odpovídají metodám aktivity. Fragment se stává zastavený, pokud aktivita, které přísluší, je stále udržovaná v zásobníku. Tento děj je znázorněn na obrázku níže.

1.5.3 Intenty a Intent filtry

Systém Android používá systém nezávislých aktivit. Pomocí intentů se můžeme z jedné aktivity dostat do jiné aktivity a předat jí data. Nejdůležitějšími částmi intentů jsou akce a předávaná data. Akce je konstanta třídy *Intent* z balíku *android.content*. Například akcí *ACTION_DIAL* s parametrem telefonního čísla lze spustit aplikaci pro vytáčení s nastaveným telefonním číslem. Intent dokáže předat objekt *Bundle*, primitivní data, jejich pole a objekty implementující rozhraní *Serializable* nebo *Parcelable*.

Existují dva typy intentů:

- Explicitní: Komponenta určená ke spuštění je identifikována pomocí jména. Používají se zejména pro spuštění jiné komponenty, pokud je dopředu znám název spouštěné komponenty (například spuštění jiné aktivity) a dopředu víme, jaká komponenta se spustí. Příklad explicitního intentu:

```
Intent i = new Intent(getApplicationContext(),
DashboardActivity.class);
```

- Implicitní: Při spouštění implicitního názvu neznáme komponentu, kterou chceme spustit, ale víme, jakou akci chceme vykonat. Například pokud chceme spustit internetový prohlížeč, vytvoříme implicitní intent ke spuštění internetového prohlížeče a systém otevře výchozí prohlížeč nastavený v mobilním zařízení. Příklad implicitního intentu:

```
Intent browserIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse(url));
```

Operační systém dává při spuštění implicitního intentu na výběr několik aplikací, kterým může intent zaslat. Aby se při určité akci nabídla naše aplikace, je nutné do souboru *AndroidManifest.xml* přidat intent filter. Ten určuje, jaké akce bude daná aplikace přijímat a jak bude nakládat se zaslanými daty. Pro každou akci musí být vytvořen vlastní intent filter, který musí být uveden v souboru *AndroidManifest.xml*.
Příklad intent filtru, převzato z [22]:

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category
android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

1.5.4 Ukládání dat

Pro trvalé ukládání data můžeme použít následující úložiště: [9]

- Shared Preferences: Jde o úložiště typu klíč-hodnota. Vhodné především pro uložení malých dat a nastavení aplikace.
- Internal Storage: Slouží k ukládání dat na sdílené vnitřní úložiště.
- External Storage: Slouží k ukládání dat na sdílené externí úložiště. Většinou se jedná o paměťovou kartu.
- SQLite Databáze: Omezená verze SQL databáze, která umožňuje ukládání komplexních dat. Databáze je reprezentována jedním souborem v interním úložišti.
- Síťové připojení neboli ukládání na server: Slouží pro ukládání dat na vzdáleném serveru, pro přístup k datům musí být zařízení připojeno k internetu. Vhodné zejména pro zálohu dat.

1.5.5 Asynchronní úlohy

Umožňují zpracovat úlohu bez vlivu na rychlost uživatelského prostředí. Uživatel tak nemusí čekat na vykonání úlohy a UI vlákno může zobrazovat jiné informace. Asynchronní úlohy tak jsou vhodné zejména pro náročné výpočetní úlohy a komunikaci se sítí. Ve skutečnosti jsou asynchronní úlohy realizovány v jednom vlákně a jsou vykonávány za sebou. [10]

1.5.6 Služby

Na rozdíl od asynchronních úloh se služby (angl. Services) používají především pro opakující se úlohy nebo náročné dlouhodobé procesy bez uživatelského rozhraní. [11]
Služby mají dva základní stavy: spuštěná a vázaná. Spuštěná služba běží zcela nezávisle na aktivitě a běží do té doby, než vykoná svou práci. Vázaný služba je svázaná s aktivitou a s jejím ukončením se služba ukončí. Typický příklad pro použití služby je stahování dat na pozadí nebo práce se soubory.
Jedna ze služeb je Alarm Manager, který má na starosti spouštět v pravidelných intervalech událost. Příkladem může být, v aplikaci implementovaný, alarm manager, který hledá blížící platby.

1.5.7 Broadcast Receivers

Broadcast Receivery, neboli posluchači událostí jsou objekty, které reagují na specifické události [12]. V operačním systému Android jsou některé broadcast receivery implementovány, ale vývojáři si také mohou vytvořit své vlastní. Příkladem může být, v aplikaci implementovaný, broadcast receiver, který při spuštěné akci výše zmíněným alarm managerem vytvoří novou notifikaci o blížící se platbě a zobrazí ji uživateli.

1.5.8 Content Providers

Svým použitím a povahou připomínají použití databáze. Zapouzdřují data a umožňují přístup ke strukturovaným datům. Poskytují rozhraní pro přístup k datům jiného procesu nebo aktivity [13].

1.5.9 Lokalizace

Operační systém Android je velmi dobře uzpůsobený pro lokalizaci aplikace. Všechny řetězce jsou ukládány v souboru *strings.xml* ve složce *res/values*. Ukázkový kód pro řetězec vypadá následovně:

```
<string name="get_location">Get Location</string>
```

Pro každý jazyk je potřeba vytvořit nový soubor *strings.xml* ve složce jazyka (například */res/values-cs*). Systém poté použije soubor dle jazykového nastavení telefonu. Jazyk je také možno dynamicky měnit přímo v aplikaci.

1.5.10 Notifikace

Slouží pro upozornění uživatele na blížící se událost, příchozí e-mail, zprávu a mnoho dalších upozornění, a to bez toho aniž by byla aplikace spuštěná. Může obsahovat malou a velkou ikonu pro snazší identifikaci v notifikační liště, titulek, zprávu a tlačítka pro interakci s uživatelem.

1.5.11 Uživatelské rozhraní

Uživatelské rozhraní je tvořeno komponentami, které lze do sebe zanořit a složit tak výsledný vzhled skládáním mnoha prvků. Každý prvek uživatelského rozhraní dědí ze třídy *View* a dále ji rozšiřují. Uveďme některé často používané komponenty:

- **Button**: tlačítko
- **TextView**: prostý text
- **EditText**: pole pro vložení uživatelského vstupu
- **ImageView**: obrázek nebo ikona

Dále OS Android umožňuje tyto komponenty skládat a seskupovat dovnitř tzv. kontejnerů. Kontejnery dědí ze třídy *ViewGroup*. Nejpoužívanější kontejnery jsou následující:

- **LinearLayout**: Podle nastavené orientace zobrazuje prvku pod nebo vedle sebe.
- **RelativeLayout**: Umožňuje vnitřním komponentám jak relativní tak absolutní pozicování.
- **TableLayout**: Pozice komponenty je určena podle souřadnic v mřížce.

1.5.12 Soubor AndroidManifest.xml

Jedná se o soubor, který musí obsahovat každá aplikace. V souboru se nachází deklarace všech aktivit, služeb, receiverů, intent filtrů a podobně. Nachází se zde také oprávnění aplikace, která se uživateli zobrazí při instalaci aplikace. Dále je zde uveden identifikátor, název a verze aplikace.

1.5.13 Soubor build.gradle

Každý projekt vytvořený v Android Studiu obsahuje jeden globální soubor build.gradle a jeden soubor build.gradle pro každý modul. V souboru se nachází konfigurace pro kompilaci projektu a jednotlivých modulů.

2 Analýza

Před samotnou implementací aplikace je nutné prozkoumat způsob ovládání mobilních aplikací. Při vytváření aplikace je vhodné navrhnout uživatelské rozhraní pro snadné a pochopitelné ovládání na mobilním telefonu. Důležitým aspektem aplikace je bezproblémový chod celé aplikace a zamezení pádům aplikace.
Nejprve bylo třeba určit, jaké běhové prostředí bude použito. Nejvhodnější bylo použití nativní aplikace v jazyce Java. Nativní aplikace umožňuje použít systémové prostředky a zabezpečit před přímým přístupem aplikace do paměti přístroje, který je častým vznikem chyb. Dále bylo nutné navrhnout způsob ukládání záznamů. Android Framework využívá k ukládání dat několik způsobů popsaných v kapitole 1.5.4.
Bylo nutné si nastudovat problematiku správy osobních financí. Předpokládejme, že každý uživatel může mít více účtů ve více bankách a potřebuje přehledně evidovat příjmy a výdaje. Jestliže uživatel chce stav účtů evidovat, může si vybrat mezi několika řešeními popsanými v kapitole 3. Pokud uživatel chce mít v evidenci i platby z minulosti, musí je všechny ručně zadat a zaevidovat.
Je důležité při implementaci aplikace zohlednit import dat ze souborů, které se dají exportovat z bankovní aplikace. Jde především o výpisy z účtů, kde jsou zaznamenány všechny pohyby na účtu. Každá banka ovšem pro export výpisů používá jiný formát dat a tak je třeba do aplikace zahrnout možnost importu pro různé formáty dat.

2.1 Kdo bude aplikaci používat

Aplikaci budou používat uživatelé, kteří potřebují evidovat své osobní účetnictví a nosit jej stále s sebou v mobilním telefonu. Je určena pro všechny osoby, ať už někdy používali nějakou pomůcku pro správu financí, nebo se teprve rozhodují o možnosti správy svých financí.

2.2 K čemu bude aplikace sloužit

Aplikace pro správu domácích financí nabízí více využití. Uživatelé mohou aplikaci používat jak pro evidenci, tak pro analýzu osobních příjmů a výdajů. Jedním z dalších možných využití se nabízí jako upozornění na blížící se platby, aplikace bude umět upozornit na opakované platby.

2.3 Jak bude aplikace fungovat

Aplikace poběží na platformě Android určené pro mobilní zařízení. K ukládání dat bude sloužit SQLite databáze. K používání aplikace je potřebný mobilní telefon s operačním systémem Android ve verzi 4.1. S přístupem k lokačním službám a datovému připojení umožní uživatelům zaznamenat vybranou polohu platby a dokáže zobrazit na mapě polohu všech plateb.

2.4 Formáty elektronických výpisů z bankovních účtů

CSV

CSV (Comma Separated Values) je jedním z nejpoužívanějších datových formátů. Hodnoty jsou odděleny čárkou nebo středníkem. Formát ale není standartní a každá banka používá jinou strukturu dat.

ABO

Standardizovaný formát ABO je specifický především pro Českou a Slovenskou republiku. Díky tomu, že každý záznam má pevně danou délku a jednotlivá pole mají vždy stejnou strukturu a délku, lze jednoduše vybrat data ze souboru. Pevný formát a délka mají i své zápory. Především pevná délka pole limituje do něj vepsat podrobnější údaje jako v případě jiných datových formátů. Formát ABO je dále rozebrán v kapitole 2.4.1.

TXT
Pro výpis z účtů některé banky používají obyčejný textový formát s vlastní strukturou dat. Zpracování souboru je tedy velmi podobné formátu CSV.

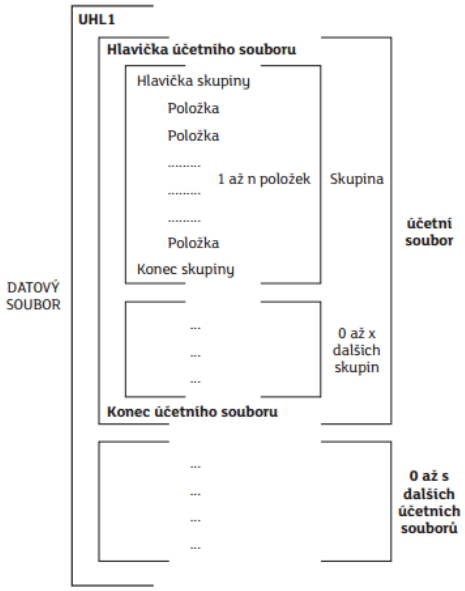
XML
Formát XML je datově nejnáročnější z pohledu objemu dat a umožňuje použít libovolnou strukturu dat. Existuje standart OFX, který zaručuje stejnou podobu dat a tak odpadává problém s nestandartním zpracováním souboru.

2.4.1 ABO formát

Formát ABO se používá především v České republice a na Slovensku. Slouží pro výměnu finančních zpráv a jeho struktura je pevně dána. Přípona souboru je .gpc. Tento typ nepoužívají všechny banky, některé upřednostňují formát txt nebo pdf. Banky, využívající tento typ formátu jsou: Fio banka, a.s., mBank S.A, Česká spořitelna, a.s., Komerční banka, a. s. a Raiffeisenbank a.s.
Export (GPC soubor) je formou elektronického výpisu banky.
Elektronický výpis má pevnou délku a obsahuje [26]:

- jeden záznam obratu pro účet a den s číslem výpisu, který je odvozen z denních výpisů
- transakce, které se vztahují ke konkrétnímu účtu a dni, transakce jsou řazeny dle čísla zpracování během zpracovávání transakcí v centrálním systému

Každý datový soubor musí obsahovat určité typy záznamu. Tato struktura je uvedena v tabulce níže.



Obrázek 3: Uspořádání datového souboru, převzato z [25]

Položka jednoduchého tuzemského platebního příkazu v CZK:

Číslo	Název	F/V	Délka min.	Délka max.	Obsah	Poznámka
1	Číslo účtu debet	V	2	17	(NNNNNN-NNNNNNNNNN)	1
2	Separátor pole	F		1	(mezera)	
3	Číslo účtu kredit	V	2	17	(NNNNNN-NNNNNNNNNN)	1
4	Separátor pole	F		1	(mezera)	
5	Částka	V	1	12	(NNNNNNNNNNN)	2
6	Separátor pole	F		1	(mezera)	
7	Variabilní symbol	V	1	10	(NNNNNNNNN)	2
8	Separátor pole	F		1	(mezera)	
9	Konstatní symbol	V	8	10	(NNNNNNNNN)	2,3
10	Separátor pole	F		1	(mezera)	
11	Specifický symbol	V	0	10	(NNNNNNNNN)	2,4
12	Separátor pole	F		1	(mezera)	
13	Zpráva pro příjemce	V	0	35	AAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAA	5
	Koncový znak zprávy	F		2	CR LF	

Obrázek 4: Struktura ABO formátu, převzato z [25]

3 Existující řešení

Před samotným začátek vytváření této aplikace jsem nejprve musel zjistit, zdali už neexistují podobné aplikace na Google play. V dnešní době existuje spousta aplikací podobného zaměření. Jde hlavně o aplikace, které slouží pro záznam příjmů a výdajů s možností zobrazení grafů.
Neexistuje však aplikace, která by dokázala předpovědět výdaje na příští období a uměla importovat záznamy z výpisů z účtu českých bank.

Wallet - Finance a rozpočet
Aplikace dokáže spravovat více účtů, zobrazit přehledy a kategorie. Je určena pro celosvětový trh a uživatelské rozhraní je dostupné také v češtině.

Expense Manager
Umí spravovat finance pro více účtů a zobrazit výpisy z daných účtů a kategorií. Aplikace je ale velmi nepřehledná a působí zastaralým dojmem. Je určena pro celosvětový trh, ale česká lokalizaci chybí.

Monefy - Money Manager
Jednoduchá aplikace, která umožňuje spravovat účty se společnou měnou. Nedokáže ukládat polohu platby a postrádá podrobnější detaily plateb.

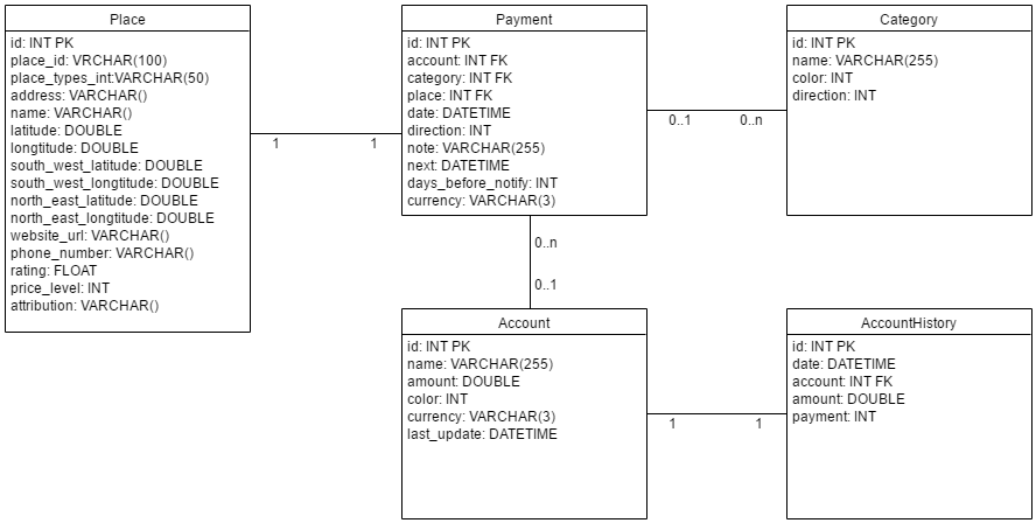
Tabulka 1: Porovnání existujících řešení

Název/ Vlastnosti	**Wallet**	**Expense Manager**	**Monefy**
Správa více účtů	Ano	Ano	Ano
Správa kategorií	Ano	Ano	Ano
Zobrazení grafů	Ano	Ano	Ano
Import z elektronického bankovnictví	Ne	Ne	Ne

4 Návrh aplikace

4.1 Návrh databáze

Při návrhu databáze bylo nutné zohlednit její použití a pokusit se o co největší optimalizaci rozsahu tak, aby aplikace neměla velké systémové nároky na práci s databází. Pro každou entitu existuje jedna tabulka, která obsahuje všechny atributy objektu, viz Obrázek 5



Obrázek 5: Návrh databáze

4.2 Výběr jazyka

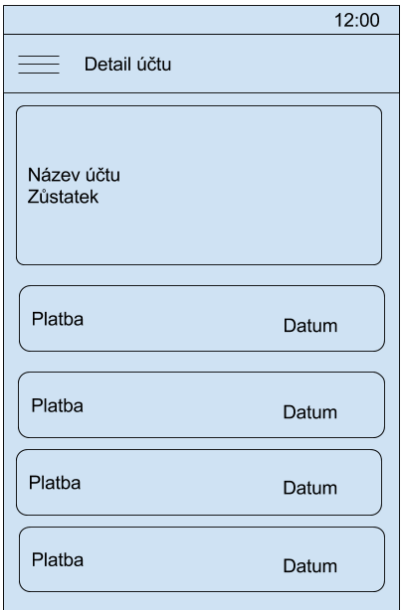
Při vytváření mé aplikace jsem si vybral programovací jazyk Java. Tento jazyk mi nabídl mnoho již vytvořených knihoven, které jsem nadále mohl využívat pro usnadnění práce.

4.3 Výběr způsobu ukládání dat

V aplikaci se bude ukládat mnoho komplexních dat, proto jsem pro ukládání zvolil SQLite databázi. Oproti jiným řešením umožňuje veškerou práci bez použití datového připojení a přístupu k externímu úložišti. SQLite databáze dovoluje ukládání velkých dat stejně jako v jiných SQL databázích.

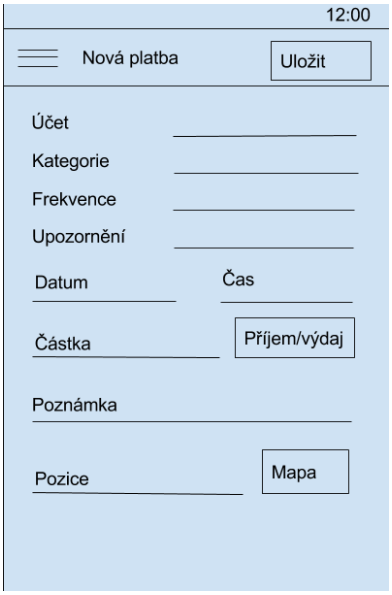
4.4 Návrh uživatelského rozhraní

Při navrhování mé aplikace jsem se zaměřil na jednoduchost a také přehlednost všech částí aplikace. Na hlavní obrazovce jsem chtěl nejvíce zdůraznit stav účtu. Také důležitou součástí hlavní obrazovky bude základní přehled výdajů a příjmů samozřejmě i s daty těchto plateb.



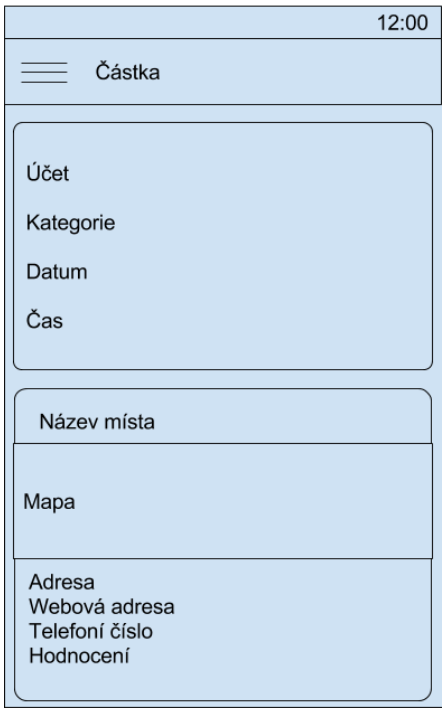
Obrázek 6:Nákres, titulní strana

Pro rychlejší orientaci bude na hlavní obrazovce umístěno tlačítko při přidání nové platby. Pro podrobnější zadávání platby, bude možnost přidat i kategorii platby. Důležitou součástí obrazovky pro novou platbu bude i možnost zvolit frekvenci opakování plateb a dále nastavení upozornění na budoucí platby a možnost vybrání kolik dní předem bude chtít být uživatel upozorněn. Výhodou bude i to, že bude možnost přidat momentální pozici.



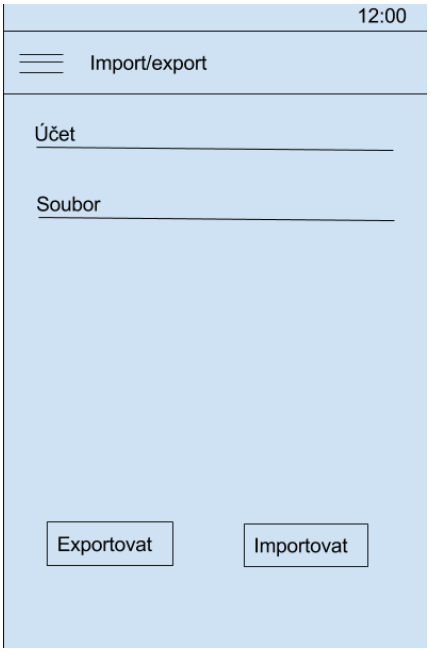
Obrázek 7: Nákres, nová platba

Po uložení platby bude možno si zobrazit zpětně celý přehled platby. V detailu se bude zobrazovat mapa platby. Pokud se jedná o platbu v obchodním řetězci či nějaké firmě, zobrazí se i webová adresa a kontakt.



Obrázek 8: Nákres, detail platby

Mou inovací je import a export výpisů z bank, proto je tato obrazovka velice důležitá. Uživatel si bude moci vybrat soubor z telefonu a následně si bude moci vybrat, jestli chce daný soubor importovat do aplikace nebo exportovat všechny platby a účty do jednoho souboru.



Obrázek 9: Nákres, import/export

4.5 Výběr nejnižší podporované verze Android API

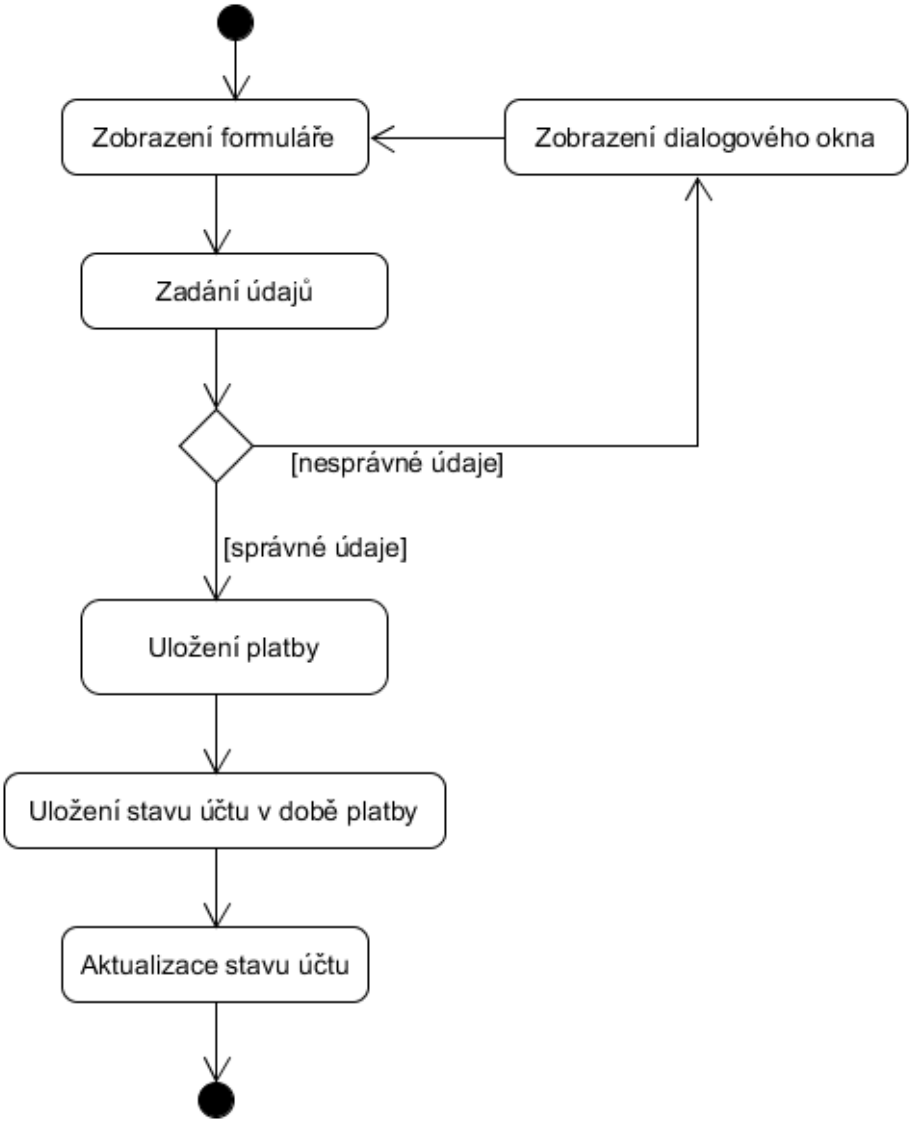
Při výběru nejnižšího podporovaného API bylo nutné zvážit přínos každé nové verze Android API a její zastoupení na trhu. Protože verze nižší než 4.1.x má pouze 4,9 % zařízení, bylo nejvhodnější zvolit verzi API 16. Ta přináší lepší správu paměti, nové části uživatelského rozhraní, optimalizaci pro televizní obrazovky a pozměněné oprávnění k práci s externí paměťovou kartou.
Použitím verze 16 je zaručena kompatibilita s, až 95 % mobilních zařízení. Android API verze 16 byla vydána na trh v červenci roku 2012, takže v době psaní této bakalářské práce je skoro 4 roky stará.

Tabulka 2: Podíl zařízení jednotlivých verzí

Verze	**Název**	**Verze API**	**Podíl na trhu [%]**
2.2	Froyo	8	0,1
2.3.3.-.2.3.7	Gingerbread	10	2,6
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2,2
4.1.x	Jelly Bean	16	7,8
4.2.x		17	10,5
4.3		18	3,0
4.4	KitKat	19	33,4
5.0	Lollipop	21	16,4
5.1		22	19,4
6.0	Marshmallow	23	4,6

4.6 Diagramy aktivit

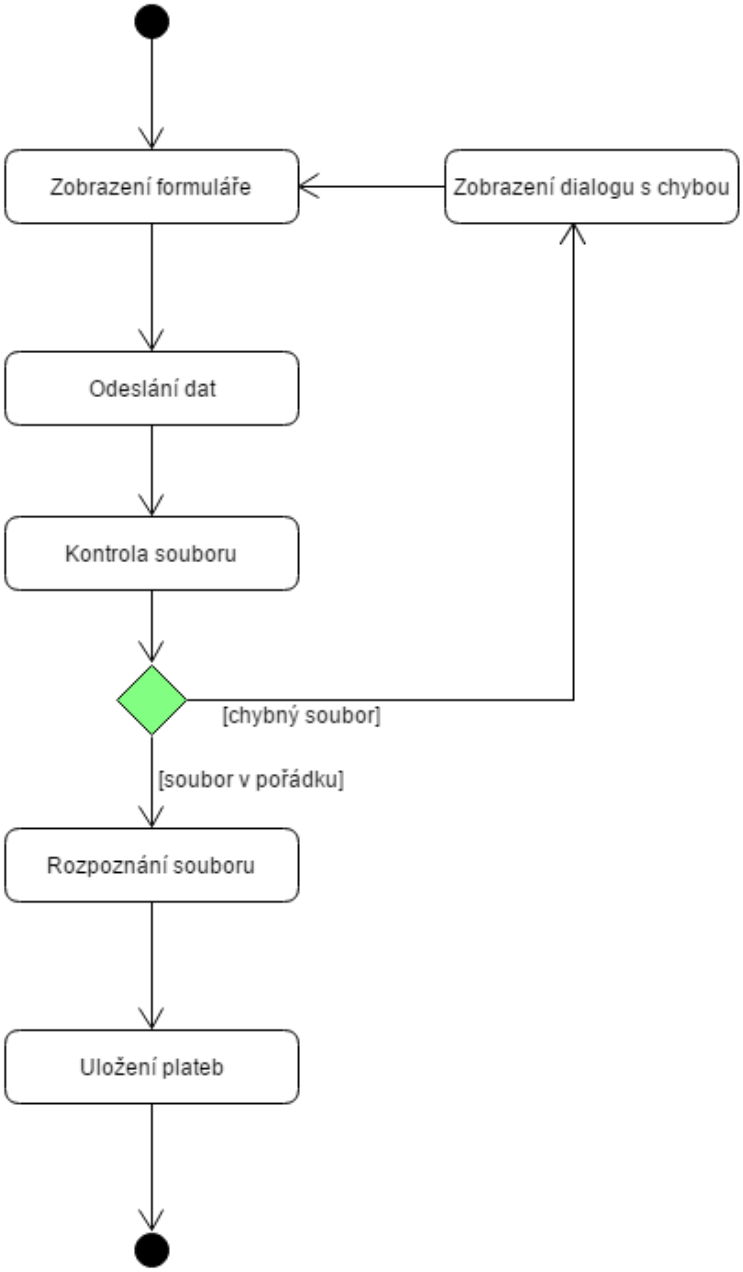
4.6.1 Nová platba



Obrázek 10: Diagram aktivit, nová platba

4.6.2 Diagram importu ze souboru

Na tomto diagramu jsem zobrazil, jak funguje import ze souboru. Můžeme zde vidět, jak aplikace pracuje, například pokud se zvolí nesprávný soubor.



Obrázek 11: Diagram importu ze souboru

4.7 Použité knihovny

4.7.1 AppCompat, Design, Support

Základní knihovny umožňující použití nových komponent také ve starších verzích operačního systému Android. Tyto knihovny také obsahují pokročilé komponenty UI.

4.7.2 Location Services

Knihovna nutná pro použití lokačních služeb a Google Places API, která obsahuje důležité komponenty pro snadné a uživatelsky přívětivé vyhledávání místa nebo adresy. Poskytuje nástroje pro získání detailních informací o zvoleném místě a poloze zařízení.

4.7.3 Gson

Knihovna Gson je vytvořena společností Google sloužící k serializaci a deserializaci objektů do formátu JSON.
JSON formát je oproti formátu XML datově méně náročnější a tedy vhodný pro přenos dat po internetu a ukládání do souboru. Pro kódování a dekódování je možné použít jakýkoliv jazyk.

4.7.4 Sugar ORM

Sugar ORM[14] je ORM wrapper pro usnadnění práce s SQLite databází. Díky této knihovně je práce s databází velmi jednoduchá. Umožňuje vazby mezi tabulkami a pokročilý výběr dat. Níže uvedený příklad ukazuje výběr 50 posledních plateb seřazených podle data sestupně:

```
Select<Payment> transferSelect = Select
        .from(Payment.class)
        .limit("50")
        .orderBy("date DESC");
```

Knihovna svým použitím usnadňuje práci s objekty v databázi a zároveň tvoří vrstvy mezi databázovými tabulkami a objektovými modely.

4.7.5 ButterKnife

Pro práci s komponenty uživatelského rozhraní je nejprve nutné komponentu najít v automaticky generované třídě R, která se nachází v balíku aplikace a generuje se při každé kompilaci projektu. V této třídě jsou uloženy všechny prvky uživatelského rozhraní ze všech *xml* souborů (barvy, styly, komponenty, řetězce a další). Každý prvek je reprezentován adresou v šestnáctkové soustavě a slouží k jednoznačné identifikaci.
Příklad adresy EditTextu s id *amount_et*:

```
public static final int amount_et=0x7f0e00b3;
```

Nalezení komponenty v aktivitě vypadá následovně:

```
EditText amoutEt = (EditText) findViewById(R.id.amount_et);
```

Knihovna ButterKnife[15] usnadňuje hledání komponenty pomocí anotace před deklarací komponenty v aktivitě nebo fragmentu. ButterKnife přetypuje danou komponentu na potřebný datový typ.
Příklad použití knihovny ButterKnife, výsledek je totožný s předchozím příkladem.

```
@Bind(R.id.amount_et)
EditText amountEt;
```

4.7.6 MaterialDrawer

Do aplikace je možné vložit Navigation Drawer. Ten však ne vždy funguje úplně správně a podle očekávání. Existuje knihovna MaterialDrawer[16], která umí sjednotit vzhled pro všechny novější verze Android API. Umožňuje snadné vytvoření Navigation Draweru a jeho vlastní úpravu.

4.7.7 Material Dialogs

Android v základu umožňuje používat dialogy. Ty jsou v analogii s počítačovými programy vyskakovací okna, která čekají na interakci s uživatelem. Stejně jako v počítačových programech může dialog v mobilním zařízení obsahovat například text, uživatelský vstup a tlačítka pro potvrzení nebo zamítnutí. Android však dovoluje do dialogu vložit téměř cokoli. V praxi se dialogy používají zejména pro zobrazení hlášky uživateli nebo potvrzení.
Pro sjednocení vzhledu do Material Designu je možné použít knihovnu Material Dialogs[17]. Velmi jednoduchým způsobem tak lze vytvořit dobře vypadající dialog ve všech novějších verzích Android API.

4.7.8 HelloCharts

Jedna z důležitých částí popisované aplikace je zobrazení grafů. Grafy umožňují velmi čitelně a přehledně zobrazit data. Knihovna HelloCharts[19] poskytuje nástroje pro vytvoření grafů. Od koláčových přes bublinové po liniové.

4.7.9 MaterialDateTimePicker

Android obsahuje komponenty DatePicker a TimePicker. Ty se ovšem v každé verzi zobrazují s jiným vhledem. Pro docílení stejného vzhledu na všech verzích operačního systém Android je možné využít jednu z mnoha knihoven pro tyto komponenty. Jednou z nich je MaterialDateTimePicker[18], která je jednoduchá a poskytuje DatePicker a TimePicker se vzhledem Material Design.

5 Implementace

V této kapitole je popsána samotná implementace a struktura projektu, rozdělení a popis tříd. Je zde vysvětlen způsob importování výpisu z účtu a taky předpověď budoucích plateb pro příští měsíc.

5.1 Import výpisu z účtu

Nejdříve je potřeba otevřít soubor a zjistit o jaký datový formát se jedná. Protože u nestandartních datových typů neexistuje předepsaná forma dat, musel jsem vytvořit pro každý formát metodu, která se o zpracování souboru postará. Nelze se spolehnout jen na příponu souboru, protože například GE Money Bank, a. s. i AirBank, a. s. používají soubory s příponou .csv, ale každá do souboru ukládá jiná a jinak řazená data. Další důvod, proč nebylo vhodné použít rozpoznávání dat jen podle přípony souboru je možné uživatelské přejmenování souboru před nahráním souboru do aplikace. Aplikace by v takovém případě nedokázala korektně rozpoznat formát souboru a import by skončil neúspěšně.
Práce s ABO formátem je pro účely automatického třídění dat výhodnější. Obsah souboru má vždy stejnou strukturu a velikost dat. Velikost dat je však zároveň největší nevýhodou tohoto formátu. Data uložená v souboru nemohou přesáhnout danou velikost, a proto se do záznamu někdy nevejde celá hodnota. Příkladem budiž poznámka klatbě, ta mnohdy označuje místo uskutečněné platby. Není tak možné z platby rozpoznat všechny detaily platby.
Před samotným zpracováním je nutné vytvořit metodu, která rozpozná typ souboru. Poté bylo možné jednotlivé řádky soubor podle vzoru roztřídit a vytvořit nové platby. Pro nejefektivnější způsob zpracování souborů jsem použil asynchronní úlohu, která běží v novém vlákně. Po importu dat se uživateli zobrazí hláška o počtu úspěšně importovaných záznamů.

5.2 Vkládání a editace plateb

Pokud bychom chtěli evidovat jen jednotlivé platby bez možnosti dalších úprav a stav účtu, stačí si ukládat jen uskutečnění platby a měnit stav účtu jen podle součtu všech plateb. Pro pokročilé zobrazení dat v podobě grafů je ale nutné si ukládat stav jednotlivých účtů v závislosti na čase. Přidal jsem do databáze tabulku **AccountHistory**, která reflektuje peněžní stav účtu s datem změny a každý její záznam je spojený s jednou konkrétní platbou. Tady ale nastává problém při vkládání zpětných dat. Vložením platby, která se s datem uskutečnění před již uloženou platbu je nutné zpětně aktualizovat zmíněné historie stavu účtů.
Ještě složitější situace nastává, pokud uživatel potřebuje změnit částku u některé z plateb. V této situaci aplikace musí nejdříve zjistit, jestli existují další uložené platby. Pokud neexistují platby staršího, nebo mladšího data, aplikace vybere z databáze požadovaný záznam, aktualizuje ho a zpět uloží do databáze. Jestliže aplikace v databázi najde starší platbu, načte si z databáze stav účtu k tomuto času a podle něj aktualizuje stav účtu k času editované platby. Při nalezení plateb s mladším datem uskutečnění aplikace postupně aktualizuje historie stavu účtu spojených s těmito platbami.

5.3 Záznam místa platby

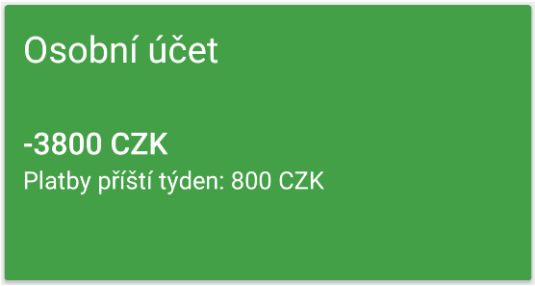
U každé platby je možnost zaznamenat polohu platby dvěma způsoby. Uživatel může místo vyhledat po kliknutí na pole pro zadání místa, nebo po kliknutí na tlačítko s ikonou pro získání polohy. Oba způsoby využívají Google Places API pro uživatelsky přívětivou volbu místa.

5.4 Upozornění na blížící se platby

Mezi atributy platby patří také možnost zvolit opakování platby. V případě, že uživatel vybere jakékoliv opakování, musí vyplnit, kolik dní předem chce být na platbu upozorněn. To je u opakovaných plateb velmi důležité, uživatel tak nezapomene například na blížící se platbu nájmu.
Aplikace v jednodenním intervalu kontroluje všechny platby v databázi, a pokud nalezne nějakou s nastaveným datem upozornění, vytvoří notifikaci o blížící se platbě.

5.5 Předpověď výdajů

Mezi užitečné funkce aplikace patří předpověď výdajů na příští období. V přehledu nebo v detailu konkrétního účtu tak můžeme vidět blížící se výdaje. Předpověď je důležitá pro případné omezení výdajů pro následující období. Výpočet předpokládaných výdajů je založen na kontrole plateb s nastavenou frekvencí.



Obrázek 12: Zobrazení karty účtu

5.6 Struktura projektu

Adresář aplikace obsahuje následující položky:

- **java/** obsahuje všechny třídy rozdělené podle balíčků (podadresářů).
- **res/** obsahuje soubory ve formátu XML sloužící pro definici stylů, zobrazení komponent aplikace, barev, řetězců apod. Jednotlivé soubory budou popsány v kapitole 5.6.1.
- **build/** obsahuje zkompilované instalační soubory s příponou apk (Application Package File), který obsahuje dex soubor (aplikace zkompilovaná do mezikódu) generované soubory potřebné ke kompilaci apod.
- **AndroidManifest.xml** slouží k základní konfiguraci aplikace. Obsahuje definici jednotlivých komponent aplikace, jako jsou například: aktivity, služby, uživatelská práva či receivery.
- **build.gradle** obsahuje konfiguraci celé aplikace. Je v něm obsažen výčet knihoven použitých v projektu, minimální požadovaná verze operačního systémů Android, pravidla pro kompilaci.

5.6.1 Struktura adresáře /res

- **drawable/** obsahuje obrázky rozdělené do podskupin (hdpi, mdpi, xhdpi, xxhdpi) podle rozlišení displeje mobilního zařízení. Ve složce jsou obsaženy především ikony a malé obrázky.
- **layout/** obsahuje xml soubory sloužící k rozmístění komponent na obrazovce.
- **menu/** obsahuje definici položek menu jednotlivých aktivit.
- **mipmap/** obsahuje pouze ikony, které se objeví na domovské obrazovce
- **values/** obsahuje následující xml soubory:
    - **colors.xml** obsahuje definice vlastních používaných barev.
    - **dimens.xml** obsahuje definici vlastních rozměrů.

    - **strings.xml** obsahuje lokalizované řetězce. Může existovat více souborů s řetězci pro jiné jazyky. Aplikace bude používat lokalizovaný soubor strings.xml podle jazykového nastavení telefonu. V případě, že se v aplikaci nenachází soubor pro požadovaný jazyk, je použit defaultní.
    - **styles.xml** obsahuje definice témat aplikace. Témata umožňují používat dědičnost a tak je jednoduché pozměnit jen část tématu.
- **xml/** obsahuje xml soubory k různým účelům. Můžeme zde uložit například položky nastavení, konfigurační datový soubor apod.

5.7 Komponenty

5.7.1 Aktivity

- **DashboardActivity** je hlavní aktivita sloužící k zobrazení přehledu účtů a přenosů.
- **AccountDetailActivity** je aktivita, které zobrazuje detail zvoleného účtu a plateb spojených s tímto účtem.
- **AccountsActivity** je aktivita pro zobrazení všech účtů.
- **ChartsActivity** slouží k zobrazení grafů pro snadný přehled výdajů a příjmů pro každý účet.
- **ImportActivity** je aktivita pro importování výpisů z účtů vybraných bank. Pomocí pomocných tříd **FileUtil** a **Parser** otevře soubor s výpisem z účtu a všechny jeho záznamy uloží do databáze.
- **MapsActivity** slouží k zobrazení plateb na mapě.
- **NewAccountActivity** slouží k vytvoření nového účtu.
- **NewPaymentActivity** slouží k zobrazení formuláře pro přidání nové platby nebo editaci již existující.
- **PaymentDetailActivity** slouží k zobrazení detailu zvolené platby, z této aktivity může uživatel platbu editovat nebo smazat.

5.7.2 Fragmenty

- **ChartFragment** zobrazuje všechny grafy pro jeden účet. Každá instance fragmentu zobrazuje jiný účet.

5.7.3 Datové třídy

Představují jednotlivé datové objekty reprezentující reálné objekty. Datové objekty mezi sebou udržují vazbu, jedná se o relační datový model.

- **Account** představuje peněžní účet.
- **AccountHistory** představuje záznam změny částky na konkrétním účtu.
- **Category** představuje kategorii platby.
- **Place** představuje lokaci platby.
- **Payment** představuje jednotlivou platbu.

5.7.4 Adaptéry

Umožňují vytvoření listu položek s vlastním zobrazením, či zobrazením komplexnějších dat. Vlastní adaptér také umožňuje použití heterogenního seznamu položek.

- **ComplexRecyclerViewAdapter** se stará o zobrazení hlavního přehledu. Zobrazuje naposledy použité účty a poslední provedené platby.

5.7.5 Broadcast Receivery

Slouží k vykonání akce na základě vyvolané události. Broadcast Receiver může reagovat například na změnu času, otočení displeje či změnu stavu připojení. Můžeme říci, že receiver může reagovat na jakoukoliv nastavenou událost.

- **RepeatedPaymentReceiver** obsluhuje událost spuštěnou AlarmManagerem. Stará se o vyvolání notifikace k blížící se opakované platbě. Po kliknutí na notifikaci se otevře detail platby s požadavkem o potvrzení platby.

5.7.6 Pomocné třídy

Tyto třídy jsou určeny hlavně k usnadnění opakovaných úkolů. Jsou zde třídy pro práci s daty a soubory.

- **FileUtil** se stará o zpracování souboru určeného k importování plateb.
- **Parser** převádí záznamy v souboru do objektů pro ukládání do databáze.

5.7.7 Asynchronní úlohy

Umožňují zpracovat úlohu bez vlivu na rychlost uživatelského prostředí. Uživatel tak nemusí čekat na vykonání úlohy a UI vlákno může zobrazovat jiné informace.

- **ImportPaymentTask** se stará o import záznamů z výpisu z účtů. Díky použití asynchronní úlohy lze importovat záznamy bez ovlivnění rychlosti hlavního vlákna.

5.7.8 Navigation Drawer

Hlavním prvkem navigace je takzvaný Navigation Drawer[21], který se vysunuje gestem po displeji z levé strany. Obsahuje odkazy na všechny důležité části aplikace a je dostupný ze všech aktivit a důležitých obrazovek aplikace.

6 Testování

Testování, potažmo ladění, je nedílnou součástí vývoje aplikace. Díky velké rozmanitosti mobilních zařízení s operačním systémem Android je velmi důležité aplikaci testovat na co možná největším množství zařízení.
Sledování aplikace a všech procesů v reálném čase nám umožňuje vestavěný nástroj DDMS a LogCat. Díky těmto nástrojů můžeme sledovat všechny události a procesy telefonu.
Při vyvíjení jakékoliv aplikace, ať už mobilní či webové, která pracuje s databází, je velmi užitečné zobrazení záznamů přímo z dané databáze. U mobilních aplikací je to trochu komplikovanější, protože každá aplikace má svou databázi v telefonu uloženo ve vnitřní paměti a běžný uživatel se k ní nedostane. Vývojář si však databázi z telefonu může stáhnout velmi jednoduše přes nástroj ADB z příkazového řádku. Například pro aplikaci com.example.app s databází pojmenovanou database.db slouží tento příkaz:

```
adb pull /data/data/com.example.app/databases/database.db
database.db
```

Databáze se z mobilního zařízení zkopíruje do adresáře, kde se příkazový řádek právě nachází. Poté je možné pro otevření souborů použít jakoukoliv aplikaci, která dokáže tento soubor otevřít.

6.1 Testování ve virtuálním zařízení

Virtuální zařízení je téměř ideální řešení pro optimalizaci aplikace pro mnoho zařízení bez nutnosti vlastnit všechny mobilních telefonů fyzicky.
To nám umožňuje správce virtuálních zařízení (AVD Manager). Při vytváření virtuálního zařízení můžeme kombinací velikosti a rozlišení displeje, výkonu, verzí operačního systému a velikosti paměti vytvořit simulaci pro téměř jakékoliv zařízení.
Jako virtuální zařízení byly použity tři emulátory, které simulovaly tyto telefony:

- Nexus 4 s verzí 4.1.1 s 1GB RAM a rozlišením 1280x768
- Nexus 5 s verzí 5.1.0 s 2GB RAM a rozlišením 1920x1080
- Nexus S s verzí 4.1.1 s 512MB RAM a rozlišením 800x480

6.2 Testování na reálném zařízení

Pro povolení testování aplikace na reálném mobilním zařízení je nejprve nutné povolit možnost ladění USB ve vývojářských nástrojích a zároveň povolit instalací aplikací z neznámých zdrojů. Pro připojení telefonu k počítači s operačním systémem Windows je také potřeba nainstalovat příslušné ovladače pro správné spojení počítače s telefonem.
Konkrétní reálné zařízení byla použita tato zařízení

- Xiaomi Redmi Note: Telefon disponuje osmijádrovým procesorem s frekvencí 1,7 GHz, 2GB RAM, obrazovkou s rozlišením 1280x720 pixelů a operačním systémem ve verzi 5.1.1.
- Doogee Dagger DG550. Telefon obsahuje taktéž osmijádrový procesor s frekvencí 1,7 GHz, 1GB RAM, obrazovku s rozlišením 1280x720 pixelů s operačním systémem ve verzi 4.4.

6.3 Chyby při testování

Při neustálém testování aplikace jsem se setkal s mnoha problémy, ať už se jednalo o mé vlastní chyby, nebo o chybný vstup uživatele. Chybám uživatelského vstupu lze částečně zamezit správným použitím atributů u formulářů. Velmi často docházelo k pádům aplikací kvůli přístupům k prázdným objektům. Tyto chyby lze díky vestavěnému nástroji LogCat velmi jednoduché identifikovat a následně opravit. Díky rozmanitosti platformy Android také bylo nutné aplikaci otestovat na více zařízeních s různým rozlišením obrazovky.

Při přechodu mezi aktivitami bylo nutné ověřovat data předávaná nové aktivitě. Mnohokrát se stalo, že nová aktivita čekala na vstupu určitá data, se kterými měla dále pracovat. Pokud data neobdržela, došlo k pádu aplikace.
Testováním jsem se snažil zajistit bezproblémový chod celé aplikace a zamezit všem možným pádům aplikace. Ty mohou nastat i v případě nedostatku systémových prostředků, což je dnes většiny telefonů spíše výjimečná situace.

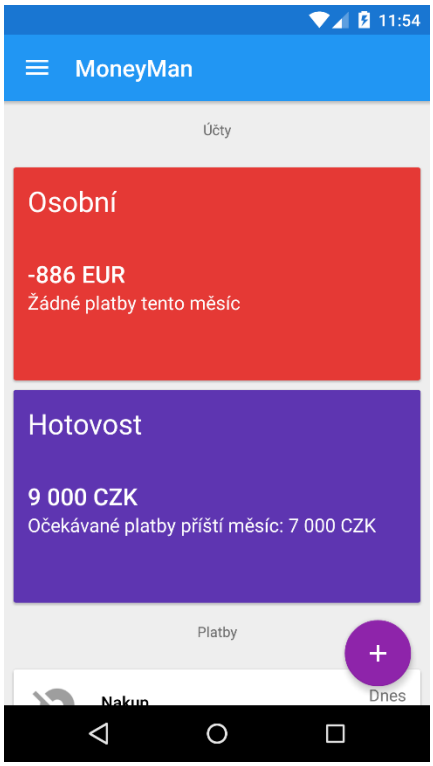
6.4 Nasazení do provozu

Otevřené operační systémy se od uzavřených liší hlavně tím, že disponují katalogem s aplikacemi pro daný systém. Do tohoto katalogu mohou vývojáři po splnění několika podmínek nahrát svou vlastní aplikaci a distribuovat ji tak potencionálním zákazníkům.
V případě operačního systému Android se jedná o Obchod Google Play. V něm můžou vývojáři nabízet jak placené, tak bezplatné aplikace. Stále rozšiřující je způsob mikro transakcí přímo v aplikaci. Ty uživateli za poplatek umožní odemknout určitou část aplikace.
K aplikaci pro publikování je možné přidat mnoho údajů. Velmi důležitý je název, popis a snímky obrazovky aplikace. Neméně důležitými údaji o aplikaci je zařazení, které může uživatelům usnadnit vyhledání aplikace.

7 Náhled uživatelského rozhraní

7.1 Hlavní obrazovka

Po spuštění aplikace vidíme na obrazovce rychlý přehled účtů a posledních plateb. U každého učtu je zobrazen název, aktuální zůstatek v měně účtu a předpokládané výdaje účtu pro příští týden. Přehled posledních plateb se nachází pod výpisem účtů a nabízí rychlý přehled o posledních proběhlých platbách.
Ve spodní části se nachází tlačítko pro přidání nové platby. Pokud se v telefonu nenachází žádný účet, tlačítko po kliknutí zobrazí formulář pro vytvoření nového účtu.
Přetažením prstu po displeji z levé části se otevře postranní menu, které slouží jako hlavní navigace v celé aplikaci.



Obrázek 13: Úvodní obrazovka

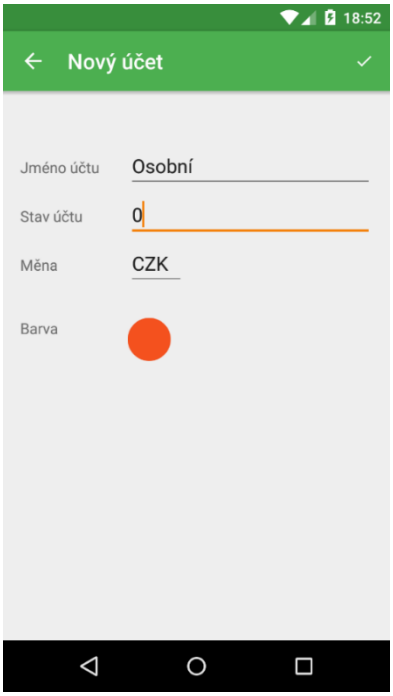
7.2 Nová platba

Na obrazovce s formulářem k nové platbě uživatel může zadat detailní údaje o platbě. Může přidat následující údaje:

- účet, ke kterému bude platba přiřazena
- kategorii platby
- frekvence opakování pro nastavení upozornění
- kolik dní předem chce být na blížící se platbu upozorněn
- čas a datum
- částku
- směr platby
- poznámku k platbě
- polohu platby, tu může buď zadat do pole, které se změní na vyhledávání míst pomocí Google Places API, nebo po kliknutí na tlačítko vybereme z nabídky okolních míst.

Pro správné uložení platby je nutné vyplnit účet, kategorii, částku a poznámku k platbě.

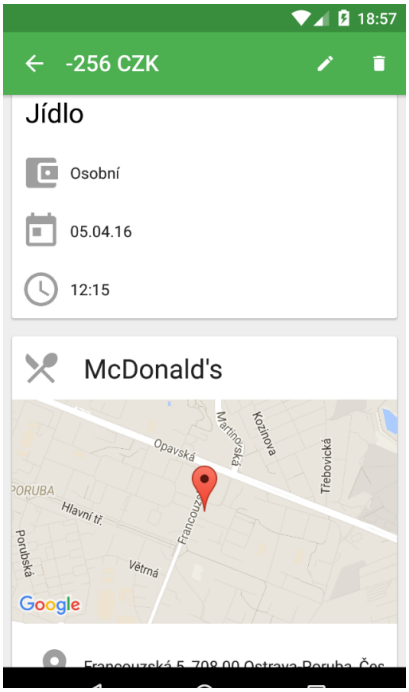
7.3 Nový účet



Obrázek 14: Vytvoření nového účtu

Při vytváření účtu si mimo názvu účtu, měny, barvy a výchozího stavu může uživatel zvolit typ banky. Tento údaj slouží především pro importování záznamů z výpisu z účtu. Pokud účet nemá nastaven typ.

7.4 Detail platby

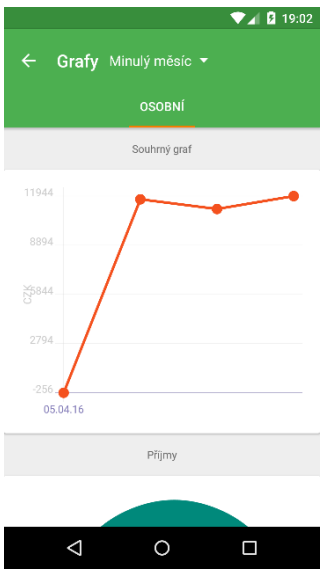


Obrázek 15: Detail platby

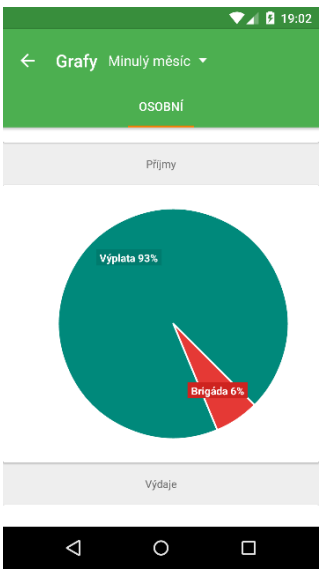
Otevřením zvolené platby se zobrazí detailní popis platby. Pokud je u platby přiřazeno místo, zobrazí se pod detailem taky mapa s informacemi o místě. Po kliknutí bod na mapě se zobrazí dvě tlačítka pro vytvoření trasy na toto místo a pro zobrazení místa na mapě v mapové aplikaci.

7.5 Zobrazení grafů

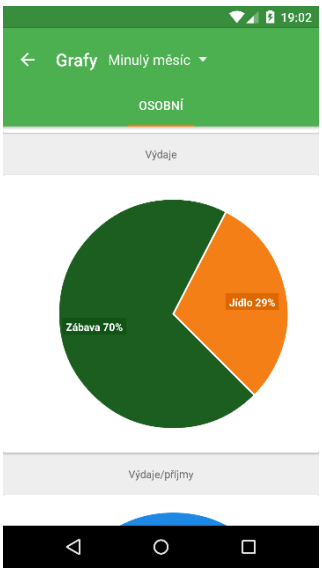
V dnešní době je dobré mít přehled o svých financích, uživatel tak může rychle vidět, za co nejvíce utrácíme a přizpůsobit tak svůj domácí rozpočet.
Pro každý účet si v aplikaci můžeme zobrazit přehledné grafy pro jednoduché a srozumitelné grafické zobrazení všech pohybů na účtu.



Obrázek 18: Souhrnný graf



Obrázek 17: Graf příjmů



Obrázek 16: Graf výdajů

7.5.1 Souhrnný graf

Tento graf uživateli umožní zobrazit příjem a výdej v jednom přehledném grafu. Pohledem na tento graf může zjistit, jak se pohyboval stav účtu během týdne, měsíce, roku nebo vše najednou. Grafické znázornění je pro uživatele mnohem přehlednější než výpis všech těchto položek dohromady.

7.5.2 Složení příjmů a výdajů podle kategorií

Cílem tohoto grafu je uživateli poskytnout přesný přehled za co byly peníze utraceny a také odkud byly peníze přijaty. Procentuální vyjádření pomůže uživateli zjistit, za co kolik utrácí a následně může pomoci při redukci výdajů podle toho, kde za poslední dobu nejvíce utratil.

7.5.3 Podíl příjmů a výdajů

Zobrazení příjmů a výdajů pomocí grafu a procentuálního zastoupení je pro uživatele další důležitá pomoc při redukci výdajů, popřípadě zvýšení výdajů. Graf ukáže, zda výdaje uživatele nejsou vyšší než příjmy.

Závěr

V rámci této bakalářské práce byla navrhnuta a implementována mobilní aplikace pro platformu Android pro správu osobních financí. Hlavním přínosem aplikace je uživatelsky srozumitelné a jednoduché importování výpisů z internetového bankovnictví. Díky aplikaci uživatelé již nemusí zadávat uskutečněné platby ručně. Aplikace tak bude díky této funkci na trhu ojedinělá a může tak konkurovat zavedeným aplikacím.
Bylo nutné nejprve prozkoumat způsob vývoje aplikací pro operační systém Android a osvojit si návrh uživatelského rozhraní. Dále bylo potřeba prozkoumat situaci na trhu a zjistit, jaké jsou stávající řešení. Uživatelské rozhraní bylo navrhnuto tak, aby bylo uživatelsky nejpříznivější pro snadnou orientaci v aplikaci.
Tak jako u každé aplikace i u této aplikace je prostor pro nové funkce a vylepšení. Jedním z mnoha je vytvoření widgetů pro zobrazení aktuálního účtu nebo poslední provedené platby na domovské obrazovce. Dále bych chtěl vytvořit uživatelské rozhraní pro tablety, tak aby tato aplikace vypadal při větších uhlopříčkách stále dobře.
Do budoucna plánuji ve vývoji aplikace pokračovat a vyvinout komplexní aplikaci pro správu domácích financí. Je nutné stále aplikaci udržovat a přinášet nové funkce a vylepšovat ty stávající. Jeden z nejdůležitějších plánů je přihlašování pomocí Google účtu a tudíž i zálohování dat na server pomocí služby Google Backup. Jelikož i bankovní systém se stále aktualizuje, proto i já bych chtěl zpravovat další typy výpisu z účtu, které jsou momentálně dostupné. Předpovídání budoucích plateb projde jistými úpravami a vylepšeními, aby mohl lépe předpovídat, kolik musí uživatel ještě zaplatit za účty. Dále chci přidat možnost načítání emailů s doručenými objednávkami a následné ukládání do aplikace. A v neposlední řadě chci optimalizovat rychlost aplikace, aby se zrychlil celý její chod. Nejsou to však všechny budoucí cíle, každá aktualizace přináší další možnosti k vylepšení stávající aplikace.

Použitá literatura

[1] Smartphone OS Market Share. IDC [online]. [cit. 2016-04-18]. Dostupné z: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[2] Google's Secret Patent Portfolio Predicts gPhone. *InformationWeek* [online]. [cit. 2016-04-18]. Dostupné z: http://www.informationweek.com/googles-secret-patent-portfolio-predicts-gphone/d/d-id/1059389

[3] Android history. *Android Central* [online]. [cit. 2016-04-18]. Dostupné z: http://www.androidcentral.com/android-history

[4] Historie a vývoj jazyka Java. *NOVOTNÝ, Luděk* [online]. [cit. 2016-04-18]. Dostupné z: http://www.fi.muni.cz/usr/jkucera/pv109/2003p/xnovotn8.htm

[5] HTML5. *W3* [online]. [cit. 2016-04-18]. Dostupné z: https://www.w3.org/TR/html5/

[6] Developing for Android - An Introduction. *Rupali Sharma* [online]. [cit. 2016-04-18]. Dostupné z: http://www.cprogramming.com/android/android_getting_started.html

[7] Xamarin platform. *Xamarin* [online]. [cit. 2016-04-18]. Dostupné z: https://www.xamarin.com/platform

[8] Support Library. Developer.android.com [online]. [cit. 2016-04-18]. Dostupné z: http://developer.android.com/tools/support-library/index.html

[9] Storage Options [online]. [cit. 2016-04-18]. Dostupné z: http://developer.android.com/guide/topics/data/data-storage.html

[10] AsyncTask. *Android Developer* [online]. [cit. 2016-04-18]. Dostupné z: http://developer.android.com/reference/android/os/AsyncTask.html

[11] Services. *Android Developer* [online]. [cit. 2016-04-18]. Dostupné z: http://developer.android.com/guide/components/services.html

[12] Broadcast Receivers. *Android Developer* [online]. [cit. 2016-04-18]. Dostupné z: http://developer.android.com/reference/android/content/BroadcastReceiver.html

[13] Content Providers. *Android Developer* [online]. [cit. 2016-04-18]. Dostupné z: http://developer.android.com/guide/topics/providers/content-providers.html

[14] Sugar ORM. Github.com [online]. [cit. 2016-04-18]. Dostupné z: https://github.com/satyan/sugar

[15] ButterKnife. Github.io [online]. [cit. 2016-04-18]. Dostupné z: http://jakewharton.github.io/butterknife/

[16] MaterialDrawer. Github.com [online]. [cit. 2016-04-18]. Dostupné z: https://github.com/mikepenz/MaterialDrawer

[17] Material Dialogs. Github.com [online]. [cit. 2016-04-18]. Dostupné z: https://github.com/afollestad/material-dialogs

[18] Material DateTime Picker - Select a time/date in style. Github.com [online]. [cit. 2016-04-18]. Dostupné z: https://github.com/wdullaer/MaterialDateTimePicker

[19] HelloCharts. Github.com [online]. [cit. 2016-04-18]. Dostupné z: https://github.com/lecho/hellocharts-android

[20] Material Design. Google.com [online]. [cit. 2016-04-18]. Dostupné z: https://www.google.com/design/spec/material-design/introduction.html#introduction-goals

[21] Navigation Drawer. Google.com [online]. [cit. 2016-04-18]. Dostupné z: https://www.google.com/design/spec/patterns/navigation-drawer.html#

[22] Intent Filters. Android Developer [online]. [cit. 2016-04-18]. Dostupné z: http://developer.android.com/guide/components/intents-filters.html

[23] Activities. *Android Developer* [online]. [cit. 2016-04-18]. Dostupné z: http://developer.android.com/guide/components/activities.html

[24] Fragments. *Android Developer* [online]. [cit. 2016-04-18]. Dostupné z: http://developer.android.com/guide/components/fragments.html

[25] Technická specifikace struktury ABO formátu. *MBank* [online]. [cit. 2016-04-18]. Dostupné z: http://www.mbank.cz/informace-k-produktum/dokumenty-ke-stazeni/dokumenty/technicka-specifikace-struktury-abo-formatu.pdf
[26] ABO formát. *Moje banka* [online]. [cit. 2016-04-26]. Dostupné z: http://www.mojebanka.cz/file/cs/bdsk_format_abo_sk.pdf

Seznam příloh

Příloha A: *Obsah přiloženého CD*

Text bakalářské práce
Zdrojové kódy aplikace
Instalační balíček aplikace